UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

AMPS Cluster: User Manual

by

Cian Davis

Materials and Surface Science Institute

University of Limerick

Version: 0.96

Colophon

This document was produced in LaTeX using TeXMaker. Graphics were produced and edited primarily using the GIMP software.

Contents

List of Figures **v**

1 INTRODUCTION **1**

1.1 Frequently Asked Questions 1

1.2 Overview 2

1.3 Conventions 2

2 THE BASICS **3**

2.1 Connecting to AMPS 3

2.1.1 Connecting a drive in Windows 4

2.1.2 Opening a command shell on AMPS 5

2.2 Basic use of the command line 7

2.3 Basic commmands 8

2.4 File locations on Linux systems 9

2.5 Editing files 11

2.6 Running graphical applications on AMPS 12

3 JOB MANAGEMENT ON AMPS 14

3.1 Overview 14

3.2 Deciding on processor usage 16

3.3 Preparing jobs for submission 16

3.3.1 FLUENT 17

3.3.1.1 Preparing files for use 17

3.3.1.2 Torque script for FLUENT 18

3.3.1.3 FLUENT Journal files and the TUI 19

3.3.1.4 Advanced use of the TUI 21

3.3.1.5 Extra FLUENT output file 22

3.3.1.6 Compiled UDFs 22

3.3.2 ABAQUS 23

3.3.2.1 Torque script for ABAQUS 23

3.3.3 NAMD 26

3.3.3.1 NAMD with Torque via charmmrun 26

3.3.3.2 NAMD with Torque via mpirun 27

3.4 Submitting a job to the cluster 28

3.5 Monitoring output from Torque 29

List of Figures

2.1 Connecting a network drive in Windows 4

2.2 Connecting a network drive for AMPS in Windows 4

2.3 Setting username and password for connecting a network drive for AMPS . . . 5

2.4 Security error from Windows when running PuTTY 5

2.5 Setting up PuTTY to connect to AMPS 6

2.6 Initial warning when connecting with PuTTY 6

2.7 Logging into AMPS with PuTTY 7

2.8 AMPS shell using PuTTY 7

2.9 Example of using the **`ls`** command 8

2.10 Example of using the **`cd`** and **`pwd`** commands 9

2.11 Files shown in Windows Explorer 10

2.12 Files shown in Linux command shell 11

2.13 Windows line ending in Linux 12

2.14 Setting the correct line endings in Notepad2 12

2.15 Enabling X11 forwarding in PuTTY 13

3.1 Torque script for FLUENT 19

3.2 Setting the RNG turbulence model using the FLUENT TUI 20

3.3 A simple FLUENT journal file 20

3.4 Starting an unsteady FLUENT simulation with the TUI 21

3.5 Advanced use of the FLUENT TUI 22

3.6 Torque script for ABAQUS 24

3.7 Torque script for NAMD using charmmrun 27

3.8 Torque script for NAMD using OpenMPI 28

3.9 Submitting a job to the queue 28

3.10 Showing the status all jobs 29

CHAPTER 1

INTRODUCTION

The AMPS cluster consists of two sets of computers dedicated to parallel solution of complex simulations. The main machines, boole and boyle, each consist of 10 IBM dual Quad-Core Intel Xeon E5430 blades with 8GB of RAM each. The older cluster, callan, consists of 9 Dell dual Pentium 4 1U servers, each with 2GB of RAM. A storage system with 9TB of diskspace supports the entire cluster.

1.1 Frequently Asked Questions

Who is entitled to an account on the AMPS cluster?
Any UL postgraduate or staff member is entitled to an account. However, time spent on the cluster will be charged at a rate depending on whether or not you are a member of the MSSI.

How do I get an account on the AMPS cluster?
E-mail cian.davis@ul.ie giving your name, department, supervisor details and what packages you will be using.

What software is available on the AMPS cluster?
Currently, FLUENT, ABAQUS, LAMMPS, OOMFF, Materials Studio (CASTEP) and C/FORTRAN compilers are available. However, if there are other packages you need, you can request them by e-mailing cian.davis@ul.ie

1.2 Overview

The AMPS system runs on Linux and allows sumbission and management of jobs from any computer in UL.

Linux systems are different to the Windows system most of us are used to. Also, most of the communication with the cluster is using the command line. While learning is not difficult, it takes a little bit of getting used to. Chapter 2 attempts to explain the differences and introduces commands which may be useful.

AMPS consists of over 20 machines. Manually scheduling who gets what machine at what time would be extremely time consuming. Instead, a queueing system is installed to automatically manage resources. This requires some extra configuration. It is explained in detail in Chapter 3.

1.3 Conventions

The manual uses several conventions in examples throughout.

`typeface` is used to denote input or output in PuTTY/command shell.

`bold` font like this denotes a command you type into PuTTY/command shell.

`italics` denotes a variable that changes, such a filename or a job number.

This flags an important point in a section. It is usually something that is easily missed. They are the items that you need to take out of a chapter and remember.

This flags a suggestion. It's not something that will cause jobs to break but it suggests best practice.

CHAPTER 2

THE BASICS

If you are not familiar with Linux, this section is extremely important. It not only explains the way to do things, but, more importantly, the concepts. In particular, using a command line is significantly different to the normal graphical view.

2.1 Connecting to AMPS

When connecting to AMPS from Windows, two steps need to be completed. First, you connect a network drive in Windows so files can be easily copied over and back. Secondly, you login remotely to AMPS so you can run commands.

Usernames are generally in the form of the first intial of your firstname and then your surname. For example, Joe Bloggs would have a username of jbloggs. In all the examples below, you should replace **jbloggs** with the username you were given when you requested an account.

2.1.1 Connecting a drive in Windows

From Windows Explorer or My Computer, click Tools -> Map Network Drive... (Figure 2.1)

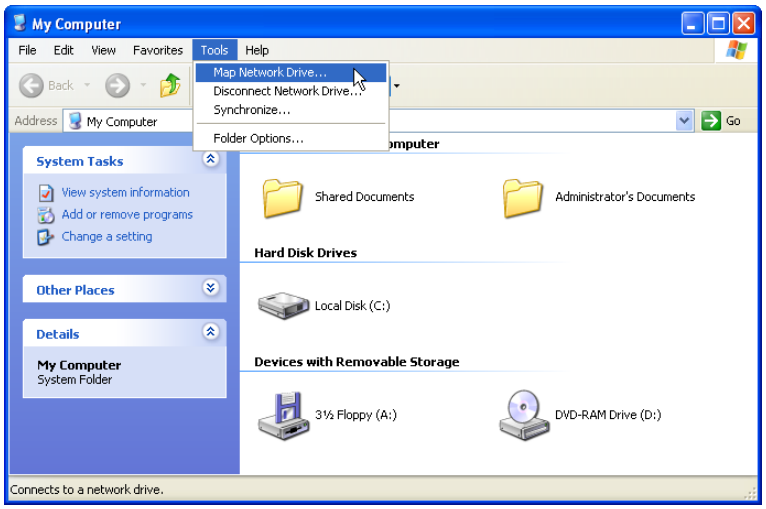

Figure 2.1: Connecting a network drive in Windows

It doesn't matter what Drive you select to connect to. The folder name is *amps.ul.campus**jbloggs* (Figure 2.2).

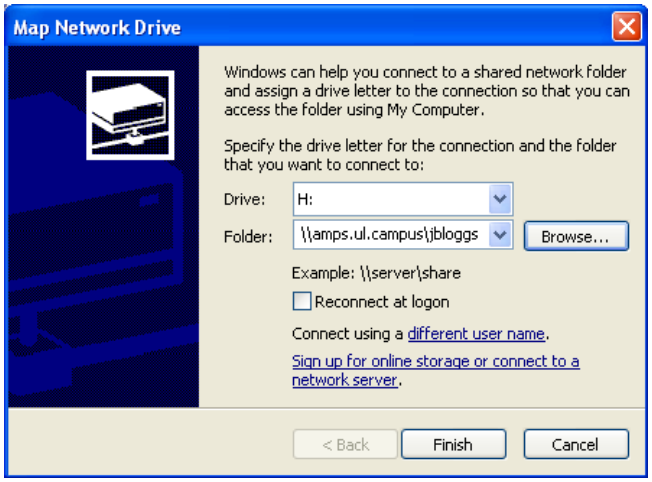
Map Network Drive
Windows can help you connect to a shared network folder and assign a drive letter to the connection so that you can access the folder using My Computer.
Specify the drive letter for the connection and the folder that you want to connect to:
Drive: H:
Folder: \\amps.ul.campus\jbloggs
Browse...
Example: \\server\share
Reconnect at logon
Connect using a different user name.
Sign up for online storage or connect to a network server.
< Back
Finish
Cancel

Figure 2.2: Connecting a network drive for AMPS in Windows

You will need to click the line that says "Connect using a different user name". Insert your AMPS username and password (Figure 2.3).

This should connect the cluster as if it was a USB drive or the likes.

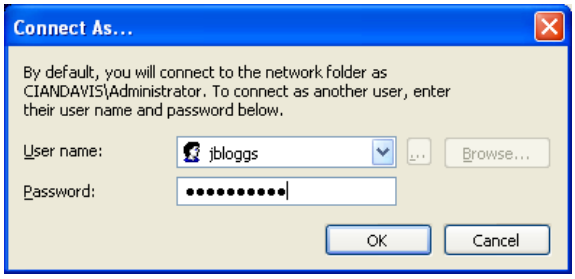
Connect As...
By default, you will connect to the network folder as CIANDAVIS\Administrator. To connect as another user, enter their user name and password below.
User name:
jbloggs
Browse...
Password:
OK
Cancel

Figure 2.3: Setting username and password for connecting a network drive for AMPS

2.1.2 Opening a command shell on AMPS

You now need to open a remote shell on AMPS by SSH using a small program called PuTTY. The best idea is to download PuTTY from http://amps.ul.campus/putty.exe and save it to your desktop. If you are outside UL, you can download it from http://www.skynet.ie/putty.exe.

Double click **putty.exe**. You may get a Security Warning from Windows stating that the publisher could not be verified (Figure 2.4). This is normal. You can prevent this warning showing every time you try and run PuTTY by unticking the "Always ask before opening this file" box.

Open File - Security Warning
The publisher could not be verified. Are you sure you want to run this software?
Name: putty.exe
Publisher: Unknown Publisher
Type: Application
From: C:\Documents and Settings\Administrator\Desktop
Run
Cancel
Always ask before opening this file
This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust.
How can I decide what software to run?

Figure 2.4: Security error from Windows when running PuTTY

Set the "Host Name" as amps.ul.campus (Figure 2.5).

If you want to save the session, enter a name in the "Saved Sessions" box and click "Save". This will then show in the large box under "Saved Sessions" every time you open PuTTY (Also shown in Figure 2.5).

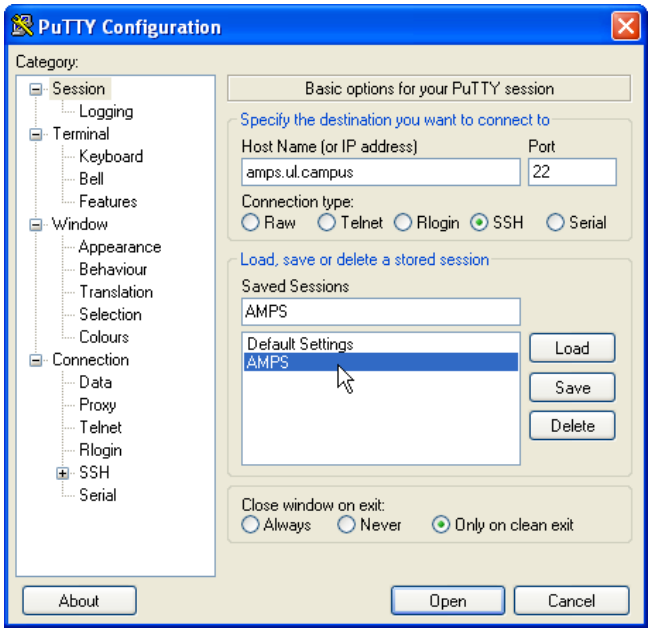

Figure 2.5: Setting up PuTTY to connect to AMPS

Click "Open" or, if you have saved your connection, double click on the name in the large "Saved Sessions" box. The first time you connect, you will receive the following warning (Figure 2.6). This is normal and you can click "Yes"

PuTTY Security Alert
The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 d0:7d:d9:34:77:ac:7d:40:ed:71:d5:3e:89:44:5e:0b
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.
Yes
No
Cancel

Figure 2.6: Initial warning when connecting with PuTTY

Enter your AMPS username at the "login as:" prompt and press enter. Then enter your password (Figure 2.7). **NOTE:** Neither your password nor stars will be shown as you type in your password.

You will then be logged into your **shell** and a **command line** will be displayed (Figure 2.8).

You can use the command line to control jobs, run system commands and get information about the system.

Figure 2.7: Logging into AMPS with PuTTY

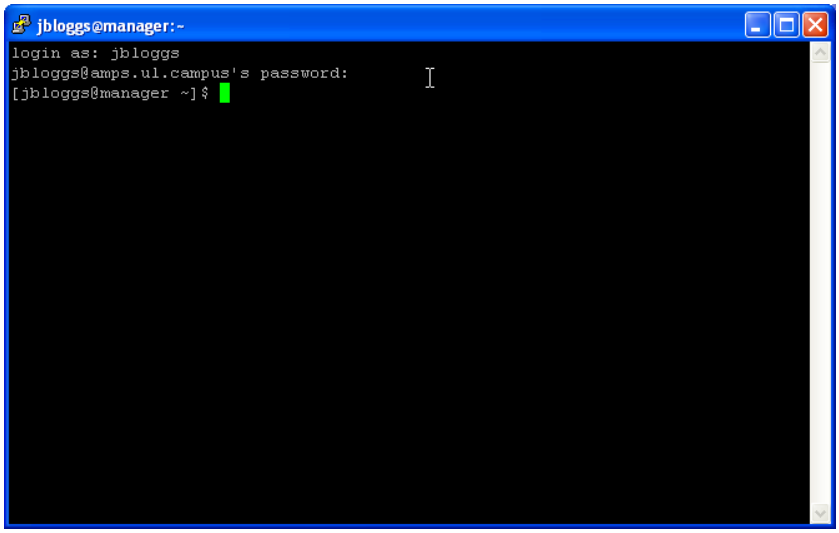

Figure 2.8: AMPS shell using PuTTY

2.2 Basic use of the command line

Graphical interfaces, equivalent to Micrsoft Windows, have been part of Linux since the start. However, the command line is an interface that is still used extensively in Linux. Because of this, it is extremely powerful.

Command history

Instead of having to retype commands, you can scroll through the history using the up and down

arrows.

Backgrounding commands
Normally, you want to be able to interact with a command. There are some commands that you want to execute in the background of the command shell. Examples are graphical applications - you interact with the graphical frontend but still want to be able to use the command line. You can do this by adding a **`&`** to the end of the line. To restore it to the foreground, use **`fg`**.

Piping
Since commands can produce a lot of output, there needs to be a way to control it. The main method is called **piping**. It allows the output of one program to be passed, or piped, to another. It is extremely easy to use. For example, if you have a lot of output, you can view it page by page by **piping** it to **`less`**. The **pipe** is | (Shift and the key to the left of Z). For example **`tail -200`** *`filename`* **`| less`**. This will take the last 200 lines of *`filename`* and show it page by page, using the **`less`** command. More information on **`tail`** and **`less`** are available in Section 2.3.

2.3 Basic commmands

Here are a few Linux commands that are useful. You type them in your PuTTY window.

`ls`
Lists files in a directory. `ls -l` will give more information. Directories will be shown in a blue coloured font.

```
[jbloggs@manager ~]$ ls -l
total 48
drwxr-xr-x 2 jbloggs users 16384 Oct 16 09:15 fluent
-rw-r--r-- 1 jbloggs users 116   Aug 18 09:53 test.txt
[jbloggs@manager ~]$
```

Figure 2.9: Example of using the **`ls`** command

`cd` *`directory`*
`cd` *`directory`* will change into the directory *`directory`*. **`cd ..`** will change to the directory below the current one (Note: There is a space between the **`cd`** and the **`..`**, which is significant. An example is shown in Figure 2.10.

```
[jbloggs@manager ~]$ cd fluent
[jbloggs@manager fluent]$ pwd
/home/jbloggs/fluent
[jbloggs@manager fluent]$ cd ..
[jbloggs@manager ~]$
```

Figure 2.10: Example of using the **cd** and **pwd** commands

pwd

`pwd` will **p**rint the **w**orking **d**irectory. This is especially useful when setting up jobs that require you to specify the location to save files or load extra subroutines. An example is shown in Figure 2.10.

tail *filename*

`tail` will print the last 10 lines of a file. It is especially useful for viewing logging output. 10 is the default but you can specify any number using `tail -`*`number filename`* and substituting in a number for *number*. `tail -f` *`filename`* will follow the output of the file as new information is written to the file. It is very useful to monitor the output of the queueing tools as they happen. You can cancel by pressing Control+C.

less

`less` is a program that will display output or a file page by page. You can scroll up and down using both the arrows keys and Page Up/Page Down. You can search by using the forward slash key (/), typing what you are looking for and pressing enter. You can repeat the same search by typing forward slash again and pressing enter.

2.4 File locations on Linux systems

File locations are slightly different on Linux systems. Linux has directories (or folders), just like Windows. Directories or folders on Windows are seperated by \ while on Linux it is /. Instead of "My Documents", Linux users have a home directory. The normal location for user files in Windows `C:\Documents and Settings\`*`username`*`\My Documents`. On Linux, it is `/home/`*`username`*.

When you connect a drive to AMPS in Windows (Section 2.1.1), you can navigate through the directories/folders just the same as you do on Windows. You can copy files to your drive on

AMPS just like you woudl copy files to a USB key. Anything that you copy into your drive on Windows will show in **/home/***username* in Linux (PuTTY).

You need to make sure that any files created on Windows but running on Linux are looking for files in the correct place and that folders/directories are created. The kind of parameters you need to check are location to auto save files, locations for user subroutines (FLUENT UDFs, Abaqus FORTRAN files) and output files.

While most utilities on Linux have support for extended characters, such as spaces, apostrophes or other punctuation marks, in file names it is a good idea not to use them. It can cause unexpected problems when using them in scripts and it is usually easier just not to use them. Underscore (_) or dash (-) are OK.

The following examples shows how the files are displayed in Windows and in a shell (PuTTY). The files in the fluent directory/folder are shown. Figure 2.11 shows the files displayed in Windows Explorer on a drive connected as described in Section 2.1.1.

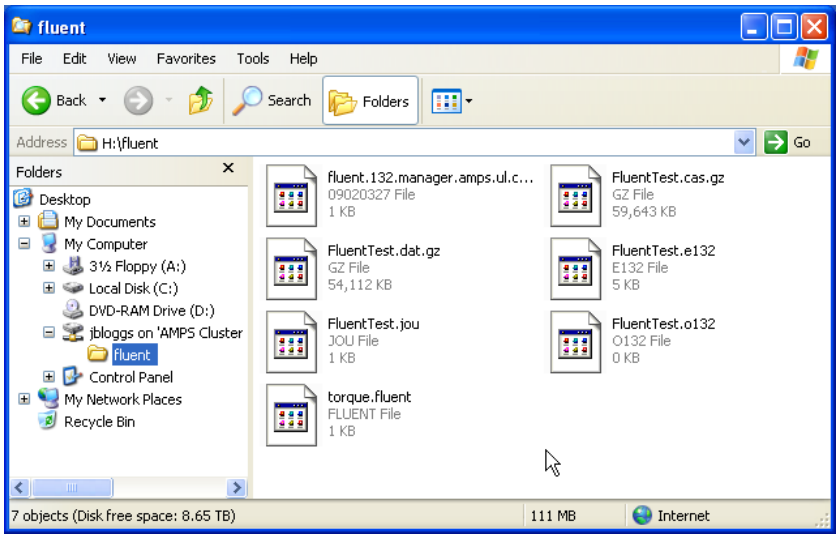

Figure 2.11: Files shown in Windows Explorer

Figure 2.12 shows the exact same folder accessed through the command line in PuTTY. The initial ~ shows you're starting off in the home directory (`/home/jbloggs`. The `ls` shows the files in the directory. The text in blue shows that `fluent` is a directory. We then change to that directory using `cd fluent/`. The / is optional. `pwd` will shows the directory you are currently in. The `ls -la` gives a detailed list of the files in the directory.

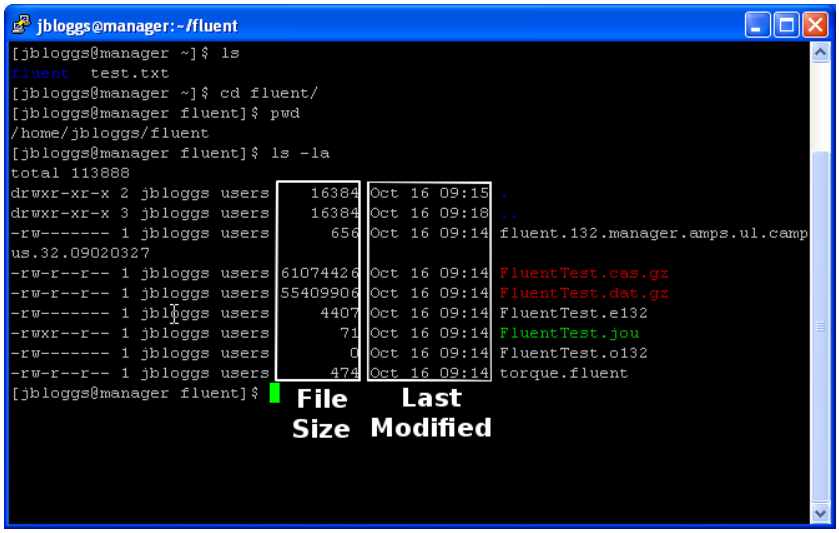
jbloggs@manager:~/fluent
[jbloggs@manager ~]$ ls
fluent test.txt
[jbloggs@manager ~]$ cd fluent/
[jbloggs@manager fluent]$ pwd
/home/jbloggs/fluent
[jbloggs@manager fluent]$ ls -la
total 113888
drwxr-xr-x 2 jbloggs users 16384 Oct 16 09:15 .
drwxr-xr-x 3 jbloggs users 16384 Oct 16 09:18 ..
-rw------- 1 jbloggs users 656 Oct 16 09:14 fluent.132.manager.amps.ul.camp
us.32.09020327
-rw-r--r-- 1 jbloggs users 61074426 Oct 16 09:14 FluentTest.cas.gz
-rw-r--r-- 1 jbloggs users 55409906 Oct 16 09:14 FluentTest.dat.gz
-rw------- 1 jbloggs users 4407 Oct 16 09:14 FluentTest.e132
-rwxr--r-- 1 jbloggs users 71 Oct 16 09:14 FluentTest.jou
-rw------- 1 jbloggs users 0 Oct 16 09:14 FluentTest.o132
-rw-r--r-- 1 jbloggs users 474 Oct 16 09:14 torque.fluent
[jbloggs@manager fluent]$
File Size
Last Modified

Figure 2.12: Files shown in Linux command shell

2.5 Editing files

Using AMPS will require you to edit some configuration files for your job. Once you have a drive connected in Windows as described above, it is simple to edit the file but a little bit of care is needed.

Windows and Linux have a slightly different way of denoting an end of a line. Windows uses two sequences whereas Linux uses one. The extra character will show up in Linux as **^M** (Figure 2.13) and will break scripts.

There are two ways to change the line endings on a file. The first is to login to PuTTY and then type **dos2unix** *filename*.

The second, recommended, way is to use a text editor that supports Linux file endings. Such a program is Notepad2, which is Free Software[1]. When you have a file open in Notepad2, click File -> Line Endings -> Unix to set the correct line endings (Figure 2.14). Notepad2 also offers some features that are very useful when writing code, such as line numbering and syntax highlighting for common computer languages, such as C (It will also show you where your `if` loops end).

[1]Free Software is software that not only costs nothing, but the code that powers it is also available. This allows a huge freedom if you like a program but want to change something about it. It is distinct in an important way from software that is free

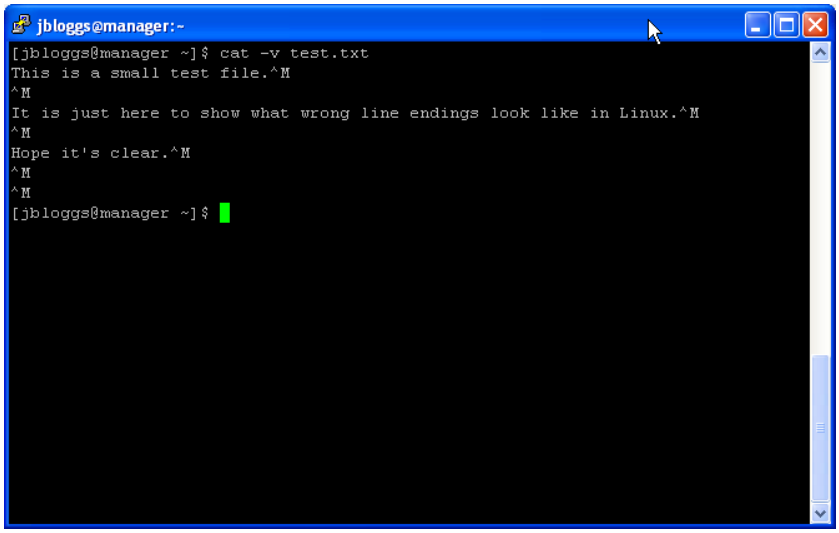

Figure 2.13: Windows line ending in Linux

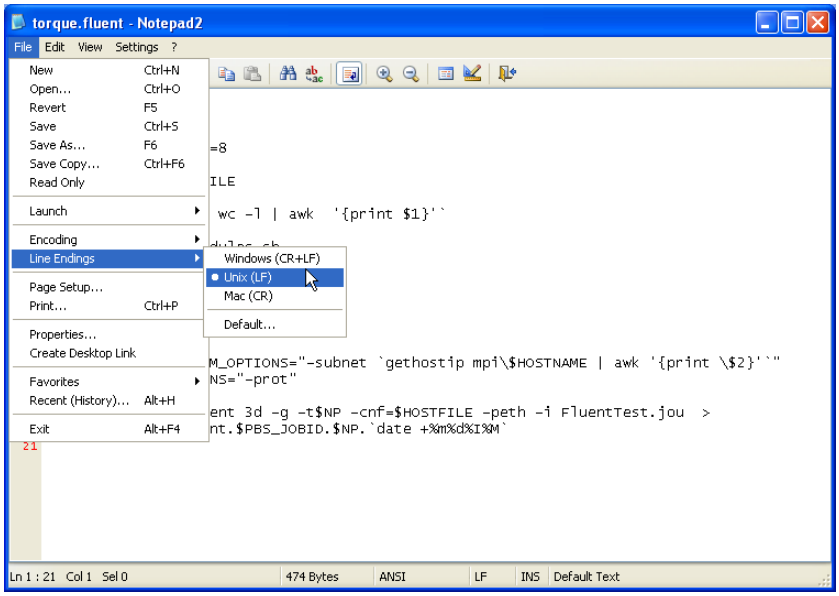

Figure 2.14: Setting the correct line endings in Notepad2

2.6 Running graphical applications on AMPS

While running graphical applications on AMPS is possible, the cluster is setup and designed for jobs to be run in the background and without graphical interfaces. Graphical interfaces should only be used for short periods (< 1 hour) in order to test applications, subroutines or settings. Use of graphical applications uses significantly more resources than normal and disrupts the

ability to queue jobs. Excessive use of graphical tools may results in a reduction in access to the AMPS cluster.

You do not **need** to use graphical applications to submit jobs. However, checking that files read correctly and that settings are correct are easier in the graphical interface. You only need this section if you want to test your files **before** you submit them to the queueing system.

Viewing graphical applications from a remote server requires use of a feature called **X-Forwarding**. While PuTTY supports this feature on Windows, an extra piece of software is needed called an X-Windows server. Exceed is a X-Windows server supported and available from ITD at $\backslash\backslash itddesktop\backslash src1\$$.

Start Exceed and then start PuTTY. Type in amps.ul.campus as the hostname as described above. Then navigate through the menu on the left to Connection -> SSH -> X11 Tick the "Enable X11 forwarding" box (Figure 2.15).

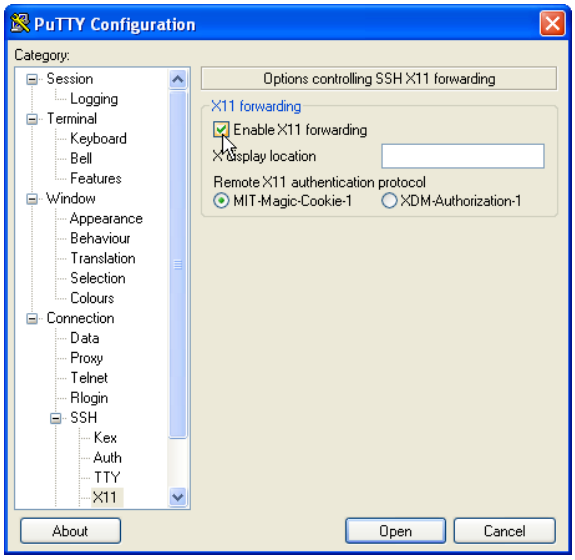

Figure 2.15: Enabling X11 forwarding in PuTTY

It is a good idea now to save the session by clicking Session, entering a name in the saved session box and clicking "Save" (Figure 2.5). Then login as normal. Now when you run graphical programs from the PuTTY command line, they will display on your screen. If you put an & at the end of command, it will run in the background and allow you to continue using the PuTTY shell for other tasks.

CHAPTER 3

JOB MANAGEMENT ON AMPS

Managing jobs and resources on AMPS is accomplished by two pieces of software. Torque manages the job queues and execution of the job on the nodes while Maui calculates the optimum way to distribute to the nodes. The scheduling system is an integral part of the AMPS cluster. With such a large number of machines and users, manually scheduling jobs would result is conflicts and reduced usage of the cluster. The scheduling system allows automatic distribution and scheduling of jobs.

However, each submission to the scheduling system requires an extra configuration file. This chapter will explain how to submit and manage jobs in the system and explain the configuration files neccesary for each piece of software available on the cluster.

3.1 Overview

Submitting a job required a few steps. This section gives a brief overview and is then detailed in the rest of the chapter. Note that every step may not be required all the time.

1. Check that your files are prepared for use on the cluster

You can do this by opening your files as normal on your own machines and checking things like Autosave locations and subroutine files. If you still aren't happy that everything is correct, you may be able to open your files on the cluster after you copy them but before you submit your job. This is specific to each piece of software and is explained further on.

2. Copy your files to the cluster

You will need to copy the files from wherever they are on your computer to the drive you connected from Section 2.1.1.

It is a good idea to create a new directory for each run so that the files do not get mixed up. So that you don't have a large number of folders, it is also a good idea to create a folder for each application you use in your connected drive. However, you need to be able to navigate around the directories on the command line (PuTTY) and know where the files you need are. Section 2.4 explains directory structure and Section 2.3 gives details on the commands needed. Particular attention should be paid to **`cd`**, **`ls`** and **`pwd`**

3. Modify the basic torque script for your application

Each application has a basic torque script, which is explained later. However, it needs to be customised with details like filenames and number of processors. This can be done from Windows.

4. Submit the job to the queueing system

Jobs are submitted to the queueing system with **`qsub`**. You also set whether you want to submit to the long or short queue. This needs to be done from your login shell (PuTTY). When you submit a job, the system returns with a job number. This is important as it is used to access all details about the submitted job.

5. Ensure job is queued correctly and monitor output

You must check that your job is not only submitted correctly but is running correctly. The job management system is setup to provide logfiles during and after a run so that you can monitor progress and find out where the problem is if things go wrong. Most of these functions are accessed from PuTTY.

3.2 Deciding on processor usage

The number of processors you request is the most basic and important choice when submitting a job. There are a number of considerations. This section gives guidelines for packages in general. Considerations specfic to each package will be addressed later in the chapter.

The first is the maximum number of processors the job can support. Multi-processing involves spliting the job between processors and solving each part individually (parallelisation). However, the method requires communication between all pieces solving the problem. The amount of communication required rises exponentially with the number of processors and becomes a significant bottleneck. This is the reason that solving a job across two processors is not twice as fast, solving across four is not four times as fast and so on. Even within software packages, certain operations parallel better and some cannot be paralleled at all. Guidelines specific to each packages are given in sections dealing with those packages.

The second, where applicable, is the number of licenses used. The AMPS cluster is the largest of its kind in UL. Since each processor usually takes a license, AMPS can easily absorb a significant proportion of licenses available in the University. Particular care should be taken with FLUENT and ABAQUS.

Each server on AMPS has dual Quad-core processors, giving 8 processors per node. If you request more than 8 processors, the number of processors per node (`ppn`) should be set to 8. If you want less than eight processors, then you must set the number of requested nodes (`nodes`) to 1. Having more than one job on a server can cause problems, slowing all jobs on the server. To optimise resources, the minimum number of servers must be used.

3.3 Preparing jobs for submission

An integral part of the queueing system is the **torque** script. It defines the number of nodes, number of processors and other job properties. Each software package requires a slightly different torque script and are explained in this chapter.

In all torque scripts, processors per node `ppn` should be set to a maximum of 8, as the AMPS servers only have 8. Setting this any higher will result in a major decrease in job speed.

It is a good idea to have a torque file in each directory you are running a job in. Not only does it have the settings you used for that particular job in case you need them in the future, but referring to files outside your current directory adds complication. The torque files can be named anything, but it's a good idea to name them consistently so they are easy to identify. In these examples, they are simply a combination of torque and the name of the application so that they can be identified quickly.

In these examples, the symbol ↩ means the line is split because it was too long but in the actual script, it should all be kept on one line.

3.3.1 FLUENT

FLUENT is a commercial CFD package. It is installed on the cluster. There are three steps that need to be completed before you run a job on the cluster. The first is checking that the case files are looking for everything in the correct place for a Linux system, the second is creating the Torque script and the last is setting up a FLUENT journal file.

3.3.1.1 Preparing files for use

This section deals purely with preparing your FLUENT files for use on the cluster. It is not part of the job submission procedure. If you are happy that everything is OK, this section can be skipped. FLUENT does not need to be open to submit a job - the queueing system will open FLUENT as it needs. However, especially for the first item of a run, it is a good idea to open files on the cluster to check everything is OK.

Changing file paths in the case files should be done as explained in Section 2.4. The specific items that should be checked are the Autosave locations, UDF paths and the location of output files.

The best way to accomplish this is to copy the case file (and data file if neccesary) over to

AMPS. You can then load FLUENT on the cluster, make the neccessary changes and save the file. You can load either the FLUENT text interface (TUI) or graphical. First, load the defaults for FLUENT.

```
[jbloggs@manager fluent]$ module load fluent
```

You can then start FLUENT (If you are using the graphical interface, please read Section 2.6).

```
[jbloggs@manager fluent]$ fluent 3d &
```

If you want to run the TUI, add a **-g** and see Section 3.3.1.3. `3d` is valid for the 3D solver. `2d` is used for 2D modules and the double precision solver for each is also available (`3ddp` and `2ddp` respectively).

3.3.1.2 Torque script for FLUENT

The standard torque script is stored in **/basic_scripts/torque.fluent**. The file is shown in Figure 3.1. You can copy it to the directory you are currently working in using **cp**:

```
[jbloggs@manager fluent]$ cp /basic_scripts/torque.fluent ./
```

While it is a long file, only the three lines hightlighted in red need to be changed.

On the first line, **FluentJob** should be substituted for the name of the case file.

On the second line, you specify the number of nodes (computers) you want to run the job on and the number of processors per node (**ppn**).

Bear in mind that the number of licenses needed is the product of the number of nodes and number of processors per node. There are only 50 licenses available in the college. Particular care should be taken during the week throughout the first semester as labs for the CFD module are running and up to 30 licenses are required.

The system does not yet automatically check for available licenses before executing the job but will do in the future.

On the third line, two changes need to be made. `3d` is valid for the 3D solver. `2d` is used for 2D

```
#!/bin/sh
#PBS -S /bin/sh
#PBS -N FluentJob
#PBS -l nodes=4:ppn=8

HOSTFILE=$PBS_NODEFILE

NP=`cat $HOSTFILE | wc -l | awk 'print $1'`

.  /etc/profile.d/modules.sh
module load fluent

export SSH_SPAWN=1

cd $PBS_O_WORKDIR

export MPIRUN_SYSTEM_OPTIONS="-subnet `gethostip mpi$HOSTNAME | awk
'{print $2}'`"
export MPIRUN_OPTIONS="-prot"

$FLUENT_INC/bin/fluent 3d -g -t$NP -cnf=$HOSTFILE -peth ↩
-i FluentJournal.jou > $PBS_O_WORKDIR/fluent.$PBS_JOBID.$NP.`date +%m%d%I%M`
```

Figure 3.1: Torque script for FLUENT

modules and the double precision solver for each is also available (`3ddp` and `2ddp` respectively). `FluentJournal.jou` should be changed to the name of the journal file you are using.

3.3.1.3 FLUENT Journal files and the TUI

A FLUENT journal file is a list of TUI commands to execute. The can be accessed in FLUENT by selecting the window and hitting enter. This will give you a list of commands that can be entered. In all cases, the commands can be abbreviated to the first three letters as long as the command in not ambigous. For commands with a hyphen in them, the first letter and each first letter after the hyphen can be used to abbreviate the command. For example `dual-time-iterate` can be abbreviated to `dti`. All commands in FLUENT can be accessed through the TUI. An example is shown in Figure 3.2 of the TUI being used to set the RNG turbulence model.

By using the TUI to find and test commands, a journal file can be easily built up. It can also automate a series of tasks. Since a job may not be immediately accessed when it is submitted to

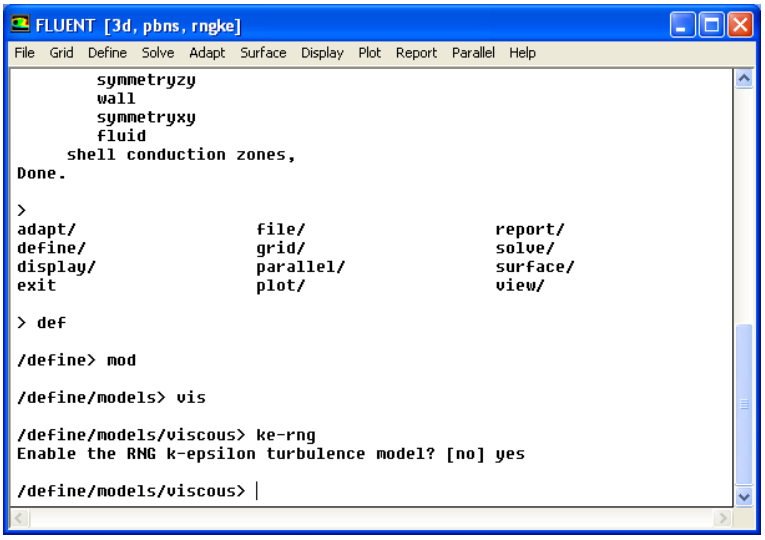

Figure 3.2: Setting the RNG turbulence model using the FLUENT TUI

the queue, the journal file provides the steps to be taken when it is executed. A simple journal file is given in Figure 3.3

```
rc FluentTest.cas.gz
/solve/initialize/initialize-flow
it 30
/exit y
```

Figure 3.3: A simple FLUENT journal file

The journal file in Figure 3.3 will read in `FluentTest.cas.gz`, initialise the flow field, solve 30 iterations and quit.

The final line in the file is extremely important. Without this, the job will not exit when completed and will clog the queue. The **y** at the end of the line will force FLUENT to exit, even if the file is not saved and is also an important requirement.

If you want to save the data at the end of the run, you need to include that in the journal file. `rc` and `it` are from a set of global commands. A full list and explanations are given in Table 3.1

Starting a unsteady simulation is slightly more difficult. It uses the command `dual-time-iterate` and is shown in Figure 3.4. It iterates for 100 time steps with a maximum of 40 iterations per time-step. In the journal file the required command would be `/solve/dual-time-iterate 100 40`. You also need to set the timestep with `/solve/set/time-step` *`time`* substituting *`time`* the numerical size of your timestep (such as 0.001).

Command	Explanation
`it`	Iterate
`q`	Quit (used to drop down a menu level)
`rc`	Read case file
`rcd`	Read case and data file
`rd`	Read data file
`wc`	Write case file
`wcd`	Write case and data file
`wd`	Write data file

Table 3.1: FLUENT TUI global commands

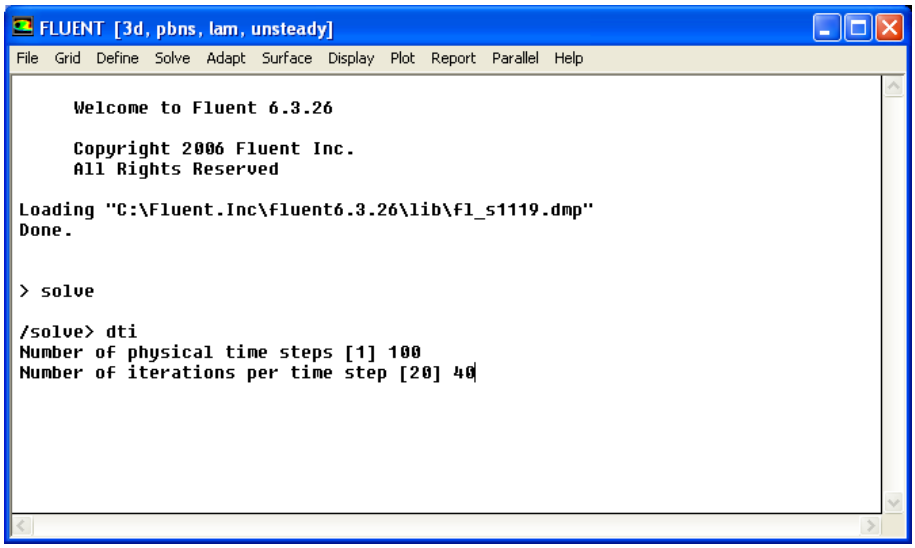
FLUENT [3d, pbns, lam, unsteady]
File Grid Define Solve Adapt Surface Display Plot Report Parallel Help
Welcome to Fluent 6.3.26
Copyright 2006 Fluent Inc.
All Rights Reserved
Loading "C:\Fluent.Inc\fluent6.3.26\lib\fl_s1119.dmp"
Done.
> solve
/solve> dti
Number of physical time steps [1] 100
Number of iterations per time step [20] 40

Figure 3.4: Starting an unsteady FLUENT simulation with the TUI

Once you have edited the Torque script and FLUENT journal, you submit the job as described in Section 3.4.

3.3.1.4 Advanced use of the TUI

Some commands in the TUI require a response from the user - such as setting up a velocity inlet. The FLUENT interface will offer standard values and show it in square brackets. When writing a journal file, you can except the standard value by using a comma (,). Many commands

will require multiple inputs and all these should be on the same line, with the command. A new line is the same as pressing enter. Figure 3.5 shows changing a mass-flow-inlet with the interactive TUI and the single line version required for a journal file.

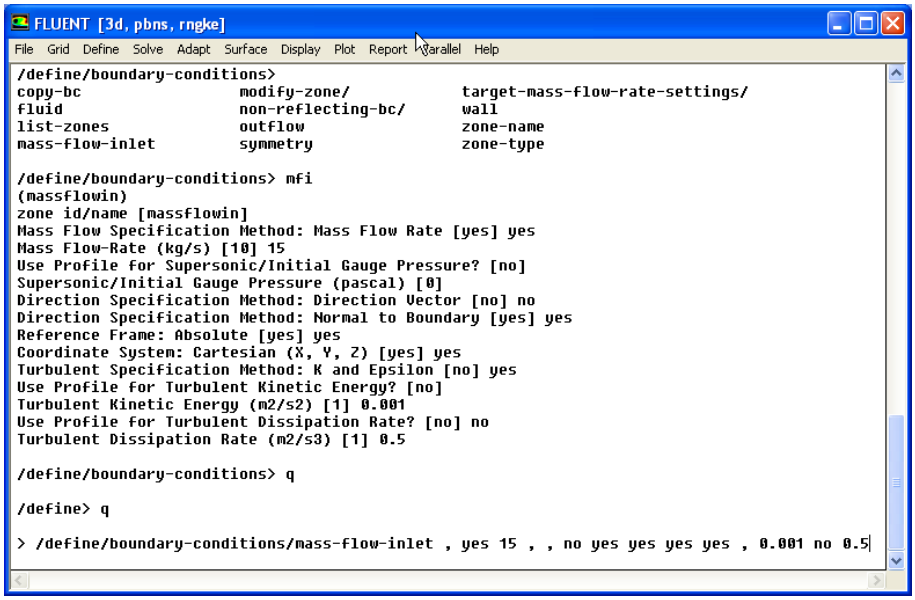
FLUENT [3d, pbns, rngke]
File Grid Define Solve Adapt Surface Display Plot Report Parallel Help
/define/boundary-conditions>
copy-bc modify-zone/ target-mass-flow-rate-settings/
fluid non-reflecting-bc/ wall
list-zones outflow zone-name
mass-flow-inlet symmetry zone-type
/define/boundary-conditions> mfi
(massflowin)
zone id/name [massflowin]
Mass Flow Specification Method: Mass Flow Rate [yes] yes
Mass Flow-Rate (kg/s) [10] 15
Use Profile for Supersonic/Initial Gauge Pressure? [no]
Supersonic/Initial Gauge Pressure (pascal) [0]
Direction Specification Method: Direction Vector [no] no
Direction Specification Method: Normal to Boundary [yes] yes
Reference Frame: Absolute [yes] yes
Coordinate System: Cartesian (X, Y, Z) [yes] yes
Turbulent Specification Method: K and Epsilon [no] yes
Use Profile for Turbulent Kinetic Energy? [no]
Turbulent Kinetic Energy (m2/s2) [1] 0.001
Use Profile for Turbulent Dissipation Rate? [no] no
Turbulent Dissipation Rate (m2/s3) [1] 0.5
/define/boundary-conditions> q
/define> q
> /define/boundary-conditions/mass-flow-inlet , yes 15 , , no yes yes yes yes , 0.001 no 0.5

Figure 3.5: Advanced use of the FLUENT TUI

When writing advanced journal files, the most common mistake is mis-counting the number of options required.

3.3.1.5 Extra FLUENT output file

FLUENT with Torque creates a third output file in addition to the two mentioned in Section 3.5. The file name is written as `fluent.JOB_ID.manager.amps.ul.campus.DATE_CODE`. For example, the file for the job shown in Figure 3.10 would be:
`fluent.73.manager.amps.ul.campus.32.08150424`.

3.3.1.6 Compiled UDFs

If you use compiled UDFs, you will need to recompile them on AMPS.

You cannot copy the compiled UDF from the Windows system to the Linux system. Also, you must recompile the UDF in a parallel solver.

To start a parallel FLUENT session on the cluster do the following:

```
[jbloggs@manager fluent]$ module load fluent
```

You can then start FLUENT (If you are using the graphical interface, please read Section 2.6).

```
[jbloggs@manager fluent]$ fluent 3d -t2 &
```

This will start FLUENT over two processors. If you want to run the TUI, add a **-g** and see Section 3.3.1.3. `3d` is valid for the 3D solver. `2d` is used for 2D modules and the double precision solver for each is also available (`3ddp` and `2ddp` respectively).

Now compile the UDF as normal and save the file with a name to signify it has a parallel UDF included.

3.3.2 ABAQUS

ABAQUS is a commercial FEA package. It is installed on the cluster. There are two steps that need to be completed before you run a job on the cluster. The first is checking that the job files are looking for everything in the correct place for a Linux system and the second is creating the Torque script.

Changing file paths in the case files should be done as explained in Section 2.4.

3.3.2.1 Torque script for ABAQUS

The standard torque script is stored in **/basic_scripts/torque.abaqus**. The file is shown in Figure 3.6. You can copy it to the directory you are currently working in using **cp**:

```
[jbloggs@manager abaqus]$ cp /basic_scripts/torque.abaqus ./
```

While it is a long file, only the four lines hightlighted in red need to be changed.

On the first line, **JOB_NAME_GOES_HERE** should be substituted for the name of the job.

On the second line, you specify the number of nodes (computers) you want to run the job on

```
#!/bin/sh
#PBS -S /bin/sh
#PBS -N JOB_NAME_GOES_HERE
#PBS -l nodes=2:ppn=4

HOSTFILE=$PBS_NODEFILE
NP=`cat $HOSTFILE | wc -l | awk 'print $1'`
cpuspernode=4

mp_host_list="["
for n in `cat $HOSTFILE`
do ; mp_host_list="$mp_host_list['$n',$cpuspernode]," ; done
export mp_host_list=`echo $mp_host_list | sed -e "s/,$/]/"`
export PATH=$PATH:/opt/intel/fce/9.1.052/bin

cat > $PBS_O_WORKDIR/abaqus_v6.env « EOF
pre_memory="16000 mb"
standard_memory="16000 mb"
standard_memory_policy=MAXIMUM
cpus=$NP
academic=RESEARCH
mp_mode=MPI
mp_host_list=$mp_host_list
mp_rsh_command='ssh -x -n -l %U %H %C'
compile_fortran="/opt/intel/fce/9.1.052/bin/ifort -c -fPIC -O0 -DLINUX -I%I"
EOF

.  /etc/profile.d/modules.sh
module load abaqus

cd $PBS_O_WORKDIR
export MPIRUN_SYSTEM_OPTIONS="-subnet `gethostip mpi\$HOSTNAME ↩
| awk 'print \$2'`"
export MPIRUN_OPTIONS="-prot"

sleep 20

abaqus job=JOB_NAME_GOES_HERE input=JOB_NAME_GOES_HERE.inp ↩
user=FORTRAN_FILE_GOES_HERE interactive

rm -rf abaqus_v6.env
```

Figure 3.6: Torque script for ABAQUS

and the number of processors per node (**ppn**). Bear in mind that the number of license tokens needed is currently calculated using equation 3.1

$$\text{Tokens} = 5 \times N^{0.422} \tag{3.1}$$

where N is the number of processors/cores you want to run on. This is the **product** of the number of nodes **and** number of processors per node.

There are only 40 license tokens available in the college. Please consider the number of licenses you use and other users in the college. Also, ABAQUS states that you will need at least 100,000 degrees of freedom per processor/core. Requesting extra processors will not speed up the solution and may even slow it down.

Table 3.2 shows how many tokens are required for a set of cores. The system does not yet automatically check for available licenses before executing the job but will do in the future. The number of processors per nodes needs to be repeated on the third line.

Cores	Tokens required	Cores	Tokens required
1	5	2	7
4	9	8	12
12	14	16	16
20	18	24	19
28	20	32	22
36	23	40	24
44	25	48	26
52	26	56	27
60	28	64	29

Table 3.2: ABAQUS token requirements for multi-core

On the fourth line, three changes need to be made. `JOB_NAME_GOES_HERE` should be changed to the name of the job (the same name as on the first line). `JOB_NAME_GOES_HERE.inp` needs to be changed to the name of the `.inp` input file. If you are using FORTRAN subroutines, change `FORTRAN_FILE_GOES_HERE` to the name of the FORTRAN file. The FORTRAN file should have a `.f` extension, though you do not include the extension on this line. If you do not use

FORTRAN subroutines, delete `user=FORTRAN_FILE_GOES_HERE` from the line.

The two lines in blue only need to be included if you are using FORTRAN subroutines. However, if they are included and no FORTRAN subroutines are used, the simulation will continue as normal.

Once you have edited the Torque script, you submit the job as described in Section 3.4.

3.3.3 NAMD

NAMD is a free molecular dynamics simulation software package.

NAMD requires a configuration file for each job. Details can be found in the NAMD documentation.

NAMD can run in parallel in two different ways. First is using `charmrun` and the second using OpenMPI. Using `charmrun`, each calculation block is solved simulataneously by all the processors assigned too it. OpenMPI will solve a single calculation block on each processor core assigned to the task.

3.3.3.1 NAMD with Torque via charmmrun

The torque script for NAMD with charmmrun is shown in Figure 3.7. While it is a long file, only the three lines hightlighted in red need to be changed.

On the first line, **`JOB_NAME_GOES_HERE`** should be substituted for the name of the job.

On the second line, you specify the number of nodes (computers) you want to run the job on and the number of processors per node (**`ppn`**). While NAMD does not have a constraint on licenses, you must be concious of other people using the cluster and the resources available. Two nodes is reasonable, four is a large block of resources and should not be requested without good reason.

The third line is the absolute path to the NAMD configuration.

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -N JOBNAME
#PBS -l nodes=2:ppn=8

CHARMM_EXEC="/opt/namd/NAMD_2.6_Linux-amd64/charmrun"
NAMD_CONFIG="/home/jbloggs/NAMD/NAMD_job.conf"
NAMD_EXEC="/opt/namd/NAMD_2.6_Linux-amd64/namd2"

export CONV_RSH="/usr/bin/ssh"
HOSTFILE=$PBS_NODEFILE
CHARMM_HOSTFILE=`mktemp -p $PBS_O_WORKDIR namd.XXXXXXXXXX`

echo "group main" > $CHARMM_HOSTFILE
for host in `cat $HOSTFILE`
do ; echo "host $host" » $CHARMM_HOSTFILE ; done

NP=`cat $HOSTFILE | wc -l | awk '{ print $1 }'`

cd $PBS_O_WORKDIR

$CHARMM_EXEC $NAMD_EXEC +p${NP} ++nodelist $CHARMM_HOSTFILE ⟵
$NAMD_CONFIG > $PBS_O_WORKDIR/namd.$PBS_JOBID.$NP.`date +%Y%m%d%H%M`

rm $CHARMM_HOSTFILE
```

Figure 3.7: Torque script for NAMD using charmmrun

3.3.3.2 NAMD with Torque via mpirun

The torque script for NAMD with mpirun is shown in Figure 3.8. While it is a long file, only the three lines hightlighted in red need to be changed.

On the first line, **JOB_NAME_GOES_HERE** should be substituted for the name of the job.

On the second line, you specify the number of nodes (computers) you want to run the job on and the number of processors per node (**ppn**). While NAMD does not have a constraint on licenses, you must be concious of other people using the cluster and the resources available. Two nodes is reasonable, four is a large block of resources and should not be requested without good reason.

The third line is the absolute path to the NAMD configuration.

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -N JOBNAME
#PBS -l nodes=2:ppn=8

MPI_EXEC="/opt/openmpi/1.2.6/gnu_4.1.2/tcp/64/bin/mpirun"
NAMD_CONFIG="/home/jbloggs/NAMD/NAMD_job.conf"
NAMD_EXEC="/opt/namd/NAMD_2.6_Linux-amd64/namd2"

HOSTFILE=$PBS_NODEFILE
NP=`cat $HOSTFILE | wc -l | awk '{ print $1 }'`

cd $PBS_O_WORKDIR

export MPIRUN_SYSTEM_OPTIONS="-subnet `gethostip mpi\$HOSTNAME | ↩
awk '{ print \$2 }'`"
export MPIRUN_OPTIONS="-prot"

$MPI_EXEC -np ${NP} -hostfile $HOSTFILE $NAMD_EXEC ↩
$NAMD_CONFIG > $PBS_O_WORKDIR/namd.$PBS_JOBID.$NP.`date +%Y%m%d%H%M`
```

Figure 3.8: Torque script for NAMD using OpenMPI

3.4 Submitting a job to the cluster

In the following examples, you should replace **`torque.script`** with the name of the Torque script you are using.

`qsub` *`torque.script`*
`qsub` submits a job to the queue. It will return the job number and domain (in our case, it will always be manager.amps.ul.campus) (Figure 3.9).

```
[jbloggs@manager fluent]$ qsub torque.script
73.manager.amps.ul.campus
```

Figure 3.9: Submitting a job to the queue

The queueing system is setup with two queues - long and short. The short queue is for jobs which will last for no longer than 2 hours and jobs in this queue will be killed after 2 hours. It is there so that short jobs do not get stuck behind long jobs for days or weeks.

By default, jobs are submitted to the short queue. To submit to the long queue, you use the

command **qsub -q long** *torque.script*.

The status of all jobs can be seen by using **qstat**. The output is shown in Figure 3.10. Time Use is the amount of processor time used and S is the state (Q is queued, R is running).

```
[jbloggs@manager fluent]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
73.manager                TestJob          jbloggs                0 R short
```

Figure 3.10: Showing the status all jobs

qdel *job.identifier*
qdel deletes a job from the queue. To delete the job shown above, you would use **qdel 73.manager.amps.ul.campus**

3.5 Monitoring output from Torque

When Torque starts, it will create two files to let you monitor the job - an output file and an error file. The output file will be named JOB_NAME.**o**JOB_NUMBER and the error file will be JOB_NAME.**e**JOB_NUMBER. For the job shown in Figure 3.10, the output and error files will be TestJob.o73 and TestJob.e73 respectively.

The best way to view these files is using **tail -f** as described in Section 2.3. This will allow you to view the output as it is created.